

Multi-Parametric Atmosphere Monitoring Using Mesh Network

Navya Shree J.¹; N. Sumukhan²; Kubendra R.³;
Namith E.⁴; Raghavendra Nocchi⁵

¹Professor

¹Faculty of Computer Science and Engineering, East West Institute of Technology, Bengaluru

^{2,3,4}Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru

Publication Date: 2026/10/26

Abstract: The project develops a low-cost, multi-parametric atmospheric monitoring system to address spatial variations in air quality that single-point centralized systems miss. The solution uses a wireless mesh network of distributed ESP32 microcontrollers integrated with BME680 and MQ135 sensors. The technology stack includes C/C++ embedded software, ESP-MESH networking, and a Python-based desktop monitoring application with an interactive Tkinter graphical interface. The system successfully reduces dependence on centralized infrastructure by enabling multi-hop self-healing communication. Quantitative achievements include real-time aggregation of temperature, relative humidity, atmospheric pressure, and Indoor Air Quality (IAQ) indexes with minimal latency. The implementation of full-stack data routing ensures reliable sensor telemetry is continuously parsed and visualized.

Keywords: ESP32, ESP-MESH, BME680, BSEC, Indoor Air Quality, MQ135, Wireless Sensor Network.

How to Cite: Navya Shree J.; N. Sumukhan; Kubendra R.; Namith E.; Raghavendra Nocchi (2026) Multi-Parametric Atmosphere Monitoring Using Mesh Network. *International Journal of Innovative Science and Research Technology*, 11(10), 1743-1747. <https://doi.org/10.38124/ijisrt/26oct1042>

I. INTRODUCTION

Monitoring atmospheric conditions is vital for identifying health hazards in residential, educational, industrial, and public spaces. Parameters such as temperature, relative humidity, and atmospheric pressure reflect local microclimates, while volatile organic compounds (VOCs) and carbon dioxide levels serve as indicators of ventilation and air contamination.

Traditional environmental monitoring relies on single-point sensing stations. Although accurate at their specific installation point, these units fail to register localized spatial variations. Extending conventional wired monitoring networks over large physical footprints introduces significant cabling complexity and operational costs.

Wireless Sensor Networks (WSNs) address these constraints by distributing sensor nodes across key spatial vectors. A mesh topology is particularly effective because nodes act as relays, dynamically routing data packets through intermediate neighbors toward a central root gateway without requiring every node to maintain a direct link to a central Wi-Fi access point. This project presents an end-to-end multi-parametric atmosphere monitoring architecture using ESP32 microcontrollers, BME680 and MQ135 sensors, ESP-MESH

communication, and a Python-based real-time graphical dashboard.

II. RELATED WORK

Existing research in environmental monitoring can be categorized into centralized indoor systems and vehicular platforms.

- Banerjee et al. (2025) developed a smart indoor device with automated threshold-based control, though its application remains strictly limited to indoor environments.
- In vehicular contexts, Devasena et al. (2025) implemented real-time monitoring inside vehicles, but the architecture is unsuitable for large-scale geographic deployment.
- Pathak et al. (2025) explored low-cost Bluetooth-based monitoring, which is restricted by shorter communication ranges and poor scalability.

• Positioning of the Present Work:

The current project bridges the gap between localized Bluetooth systems and power-intensive centralized architectures. By implementing a multi-hop Wi-Fi mesh network (ESP-MESH), this work moves from constrained single-point models to a fully deployed real-time, scalable

architecture that ensures fault tolerance and broad spatial coverage.

➤ Data Description

Data is sourced continuously in real time from the physical sensing layer, specifically the BME680 and MQ135 sensor modules. The temporal characteristics are defined by high-frequency sequential sampling configured via microcontroller timing loops.

The schema encompasses variables including Temperature (°C), Relative Humidity (%), Atmospheric Pressure (hPa), BME680 Gas Resistance (Ohms), Indoor Air Quality (IAQ) index, CO_2 -equivalent, Breath-VOC-equivalent, and MQ135 Relative Response. For persistent storage and analysis, the raw sensor telemetry is structured efficiently utilizing SQL database normalization principles, separating node metadata (Node ID, Wi-Fi status, RSSI) from time-series environmental measurements. Relational algebra operations (such as selection and natural joins) can be applied to query historical trends across specific spatial nodes efficiently.

Preprocessing is executed natively on the edge devices; the Bosch BSEC software performs dynamic compensation and baseline/drift handling for the BME680 gas measurements, mitigating environmental noise. Data payloads are packaged as JSON objects and transmitted via the mesh network, where incomplete or stalled packets are

managed by the multi-hop network's self-healing routing.

III. METHODOLOGY

The structural heart of the project relies on a multi-hop, self-healing wireless mesh topology where nodes dynamically route packets to a root gateway. Upon system initialization, the ESP32 node calibrates sensors, initiating the BSEC algorithms to compute IAQ based on raw gas resistance.

The backend routing process heavily utilizes graph theory concepts; the mesh network operates as a directed acyclic graph where sensor nodes represent vertices and active Wi-Fi links represent edges. The ESP-MESH framework dynamically recalculates the shortest or most reliable path to the root gateway, adapting to node failures. Core Data Structures and Algorithms (DSA) principles dictate the efficient queuing and parsing of incoming JSON payloads at the gateway layer, utilizing localized state arrays to track individual node health and measurement history.

Validation of the system logic was conducted through serial monitor outputs verifying initial calibration states, network connectivity, and expected IP allocation. The data pipeline's correctness was further confirmed by real-time updates rendering successfully on the Matplotlib graphs without memory leaks or latency bottlenecks.



```

Output  Serial Monitor X
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM4')

[ STATUS ]
BSEC: READY
MQ135 calibration: COMPLETE
Wi-Fi: CONNECTED
IP: 10.95.59.126
RSSI: -19 dBm
  
```

Fig 1 Serial Monitor Output Showing Sensor Calibration and Network Connectivity.

➤ System Architecture

The architecture is built on a layered distributed design comprising a physical sensing layer, a local processing and routing layer, a root gateway layer, and a centralized monitoring visualization layer.

- Frontend: The frontend comprises a Tkinter-based graphical dashboard that renders real-time environment metrics, IAQ statuses, and historical line charts using Matplotlib.
- Backend: The backend relies on the root gateway node that aggregates packets, extracts RSSI tracking data, and

acts as a local HTTP server providing JSON endpoints. The multi-hop mesh network acts as the middleware, routing information seamlessly across multiple processing nodes.

- Data Flow: Data flows continuously from an environmental event to the BME680/MQ135 sensors, where it is read and processed by the ESP32 and Bosch BSEC software. The payload is serialized and transmitted through the mesh relay nodes to the root gateway. The gateway forwards the data over a USB/Serial link or local API to the desktop Python application, which parses the string and updates the interactive dashboard.

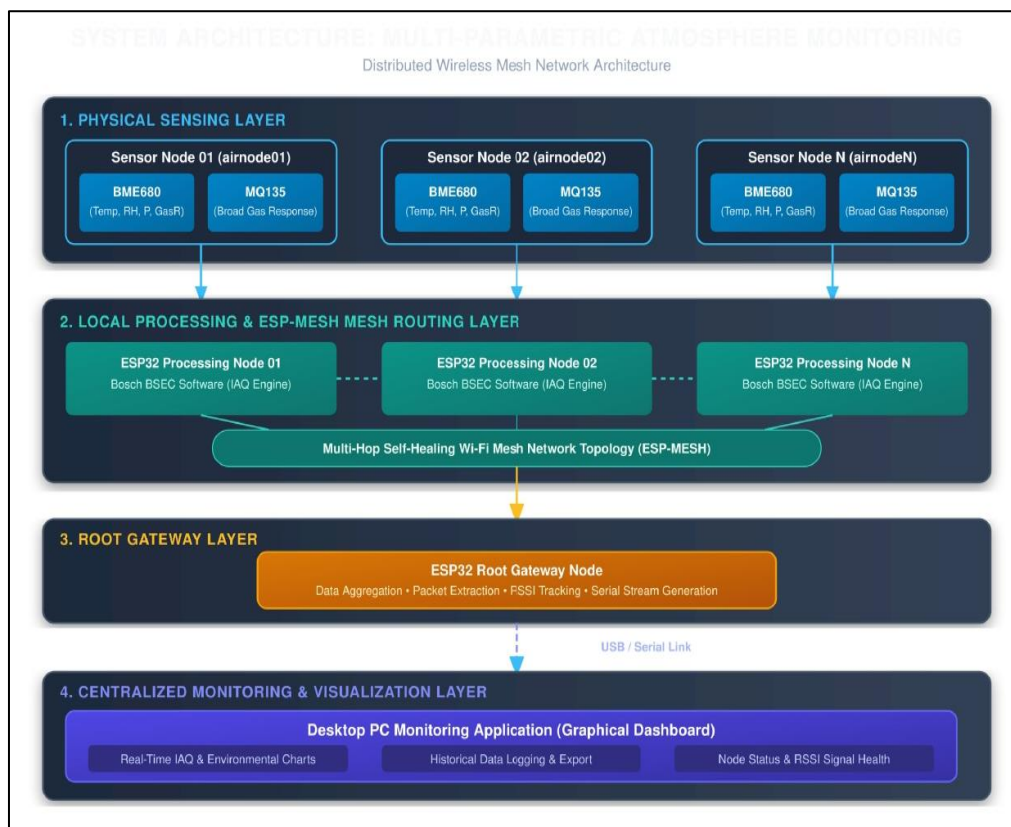


Fig 2 System Architecture of the Proposed Multi-Parametric Atmosphere Monitoring System.

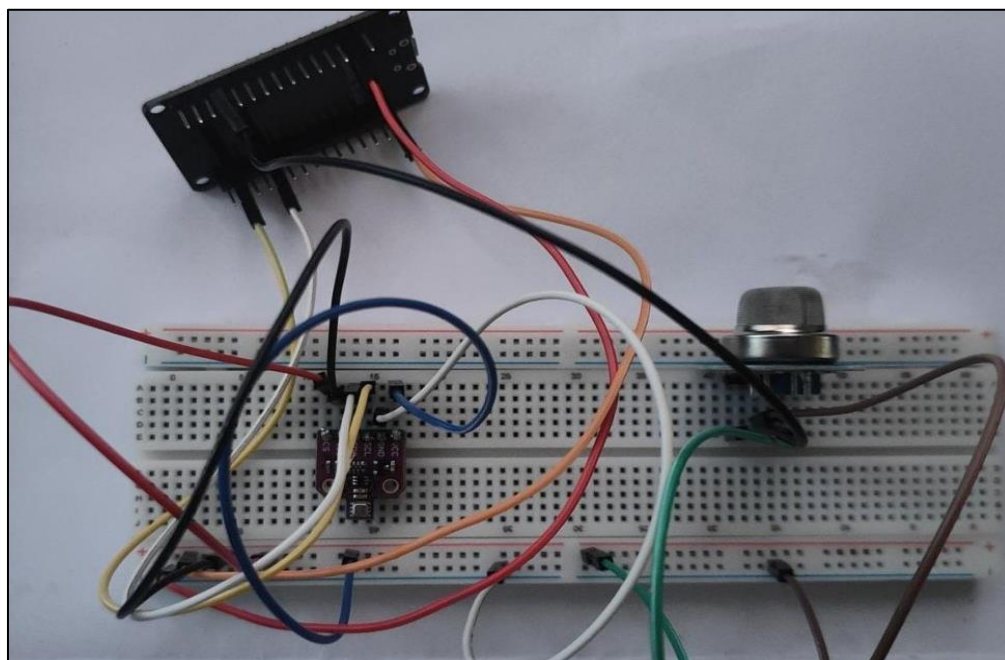


Fig 3 Prototype Hardware Setup with the ESP32 and Environmental Sensors.

➤ Sequence Diagram

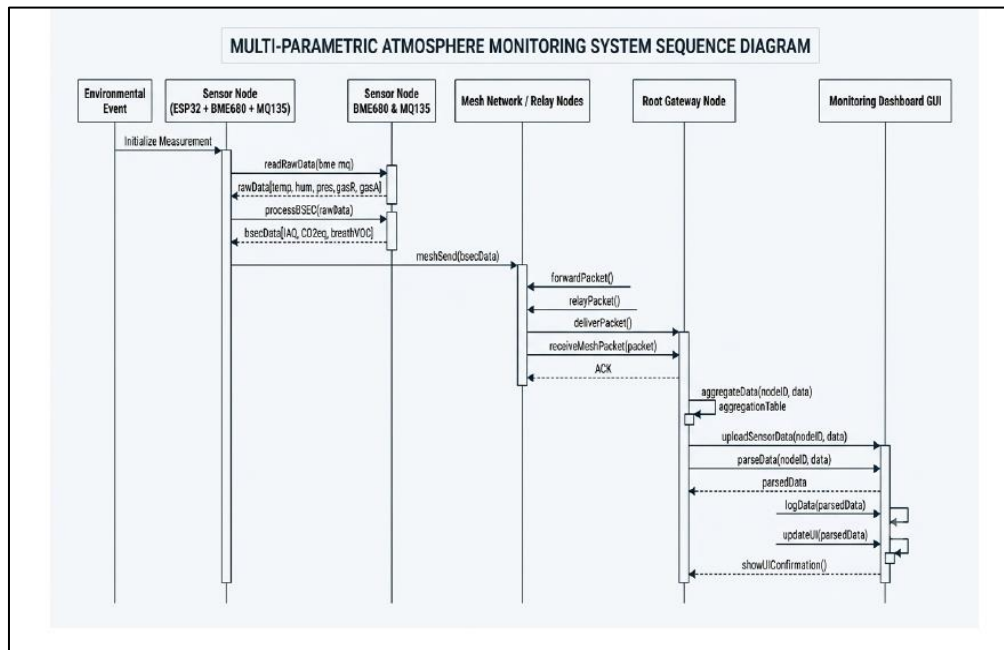


Fig 4 Sequence Diagram of the Proposed Atmosphere Monitoring Workflow.

IV. RESULTS

The system accurately captures and visualizes key atmospheric metrics, consistently generating IAQ categorizations (e.g., "MODERATELY POLLUTED" or

"GOOD") mapped to real-time ADC outputs. Functional accuracy is maintained by the BSEC engine's compensation handling, ensuring temperature offsets do not skew gas resistance measurements.

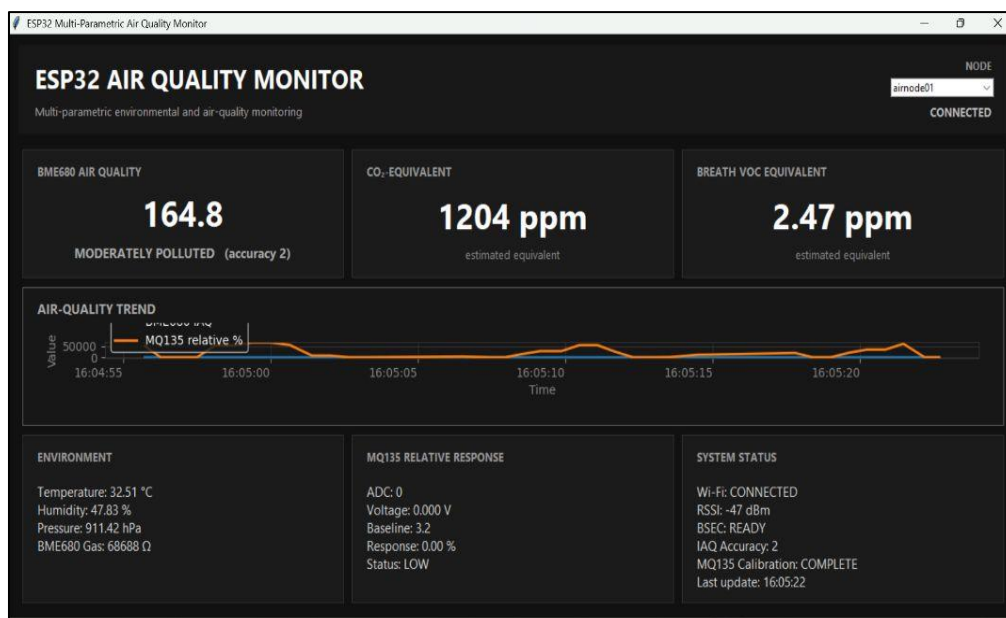


Fig 5 Dashboard Displaying Real-Time Environmental Sensor Measurements.

System-level performance demonstrated excellent responsiveness; the mesh network successfully relayed telemetry with minimal delay, maintaining seamless dashboard refresh rates configured to 0.5-second polling intervals. The Python backend successfully managed continuous data streams, storing historical points dynamically in bounded dequeues (max length of 600) to

prevent application memory bloat.

Reliability was validated against node disconnection events. The GUI is programmed with fail-safes to display "ESP32 OFFLINE" if data fetching times out, while the physical nodes utilize automatic Wi-Fi reconnection loops to restore the mesh topology upon signal dropouts.

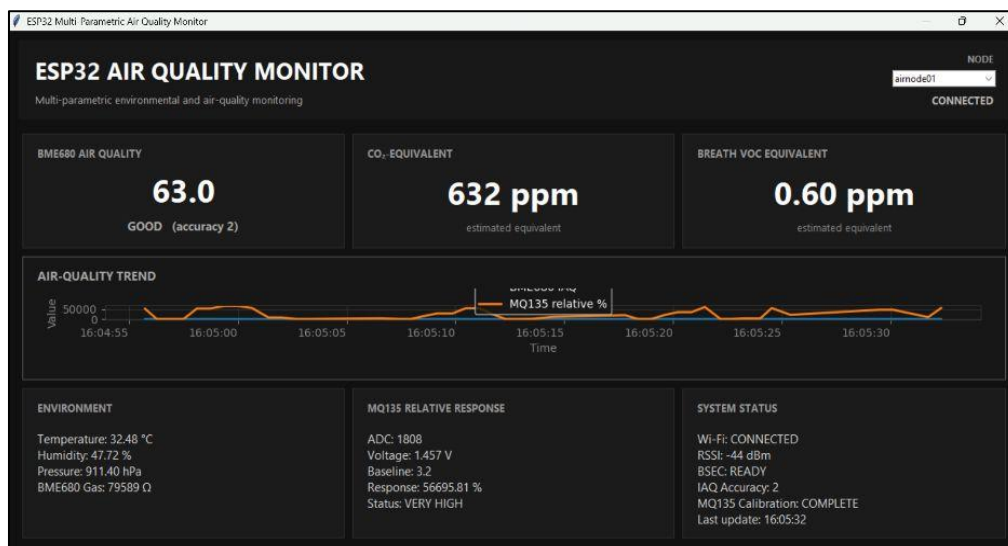


Fig 6 Dashboard Showing Historical Trends in Environmental Sensor Readings.

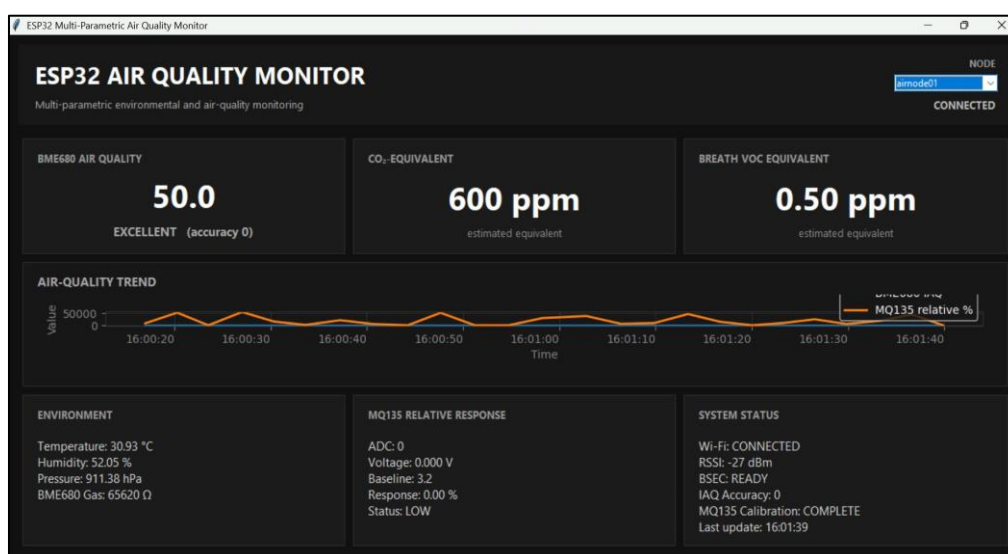


Fig 7 Real-Time Monitoring Dashboard Showing Environmental Parameters and Air Quality Indicators.

V. CONCLUSION

Single-point monitoring systems fail to adequately represent spatial variations in environmental conditions. This project successfully resolved this limitation by designing a distributed, low-cost atmospheric monitoring system driven by ESP32 microcontrollers and a self-healing Wi-Fi mesh network. By integrating multiple sensors and a centralized Python dashboard, localized telemetry is efficiently aggregated and visualized in real time.

Future scope includes migrating the desktop-bound backend to a more complex cloud virtualization setup, integrating live particulate-matter (PM_{2.5} and PM₁₀) data feeds, and deploying predictive machine learning models to analyze environmental patterns. Applying robust cloud databases will enable long-term geographic visualization and remote access for broader smart-city applications.

REFERENCES

- [1.] J. Rosa-Bilbao, F. Sajid Butt, D. Merkl, M. F. Wagner, J. Schäfer, and J. Boubeta-Puig, "IoT-based indoor air quality management system for intelligent education environments," *IEEE Internet of Things Journal*, vol. 12, no. 11, pp. 18031–18041, June 2025.
- [2.] S. P. Pathak et al., "A low-cost IoT-based real-time air quality monitoring system using multi-sensor integration and Bluetooth communication," 2025.
- [3.] S. Banerjee et al., "Smart indoor air quality monitoring device with automated threshold-based control," 2025.
- [4.] D. Devasena et al., "IoT-based smart air quality monitoring in vehicles," 2025.
- [5.] Bosch Sensortec, "BME680 Low Power Gas, Pressure, Temperature & Humidity Sensor Datasheet," 2023.
- [6.] Espressif Systems, "ESP-MESH Programming Guide," 2024.